We claim:

Sub B' 1) A process for transmitting a packet having a header and variable length payload on a communications interface comprising the steps:

a first step of sending IDLE symbols until a synchronization time has passed;

a second step of sending said packet including a START symbol and TYPE field identifying the format of said payload including an FCS sequence;

a third step of sending said variable length payload;

a fourth step of sending a terminator including an END symbol indicating end of transmission of said packet;

a fifth step of sending IDLE symbols if next said packet is not ready to transmit, or returning to said second step if said next packet is ready to transmit.

2) The process of claim 1 wherein said TYPE field uniquely identifies said payload format, said format including Ethernet packets, native IP packets, ATM cells, and control packets.

Sub A' 3) the process of claim 2 wherein said header further includes declaration fields for at least one of BPDU, PRIORITY, VLAN_ID, and an application specific field.

Sub B'

4) The process of claim 3 wherein said BPDU field is 1 bit in size.

5) The process of claim 3 wherein said PRIORITY field is 3 bits in size.

6) The process of claim 3 wherein said VLAN_ID field is 12 bits in size.

7) The process of claim 3 wherein said application specific field is 32 bits in size.

8) The process of claim 3 wherein said header comprises, in sequence, said START symbol, said BPDU field, said TYPE field, said PRIORITY field, said VLAN_ID field, and said application-specific field.

9) The process of claim 1 wherein a plurality n of data lanes carry said header, said payload, and said END symbol.

Sub A2

10) the process of claim 9 wherein

said second step comprises transmitting said header across said n data lanes until all said header information has been sent;

said third step comprises transmitting said variable length payload, wherein during a final payload cycle, said payload ends on said data lane m;

for the case where m < n, said fourth step includes sending on said final payload cycle said END symbol on lane m+1, and said IDLE symbol on any available data lanes m=n+2 through n;

for the case where m = n, said fourth step comprises sending said END symbol on said data lane 0, and said IDLE symbol on said data lane 1 through said data lane n.

11) The process of claim 10 where n = 8.

12) The process of claim 10 where n = 4.

13) The process of claim 10 where n = 2.

14) the process of claim 9 where n = 1, and

said second step comprises transmitting said header on said data lane until all said header information has been sent;

said third step comprises transmitting said variable length payload on said data lane,

said fourth step comprises sending said END symbol on said data lane.

15) The process of claim 10 wherein at least one said data lane comprises a serial electrical link.

16) The process of claim 10 wherein at least one said data lane comprises a parallel electrical link.

17) The process of claim 10 wherein at least one said data lane comprises one or more serial or parallel optical links.

18) The process of claim 10 wherein said first step comprises the transmission of said IDLE symbols on all said n data lanes.

SuBB'

19) The process of claim 18 wherein said IDLE symbols are transmitted across all said n data lanes when there is no said packet data available to transmit.

20) The process of claim 19 wherein successive data lane cycles toggle successively between the states odd and even.

21) The process of claim 20 wherein said IDLE symbols transmitted comprise IDLE_ODD symbols during said odd state, and IDLE_EVEN symbols during said even state.

22) A communication interface comprising n data lanes, said interface sequentially transmitting a header distributed across a plurality of said data lanes, a variable amount of payload data distributed across a plurality of said n data lanes.

23) The communication interface of claim 22 wherein said transmission of said header includes transmitting a START symbol on first said data lane, and the transmission of said payload data is followed by an END symbol on at least one said data lane.

24) The communication interface of claim 23 wherein said transmission of said payload data includes transmitting data across said n data lanes up to data lane m, where m <= n.

25) The communication interface of claim 24 wherein if said m < n, said END symbol is transmitted on data lane m+1, and if said m=n, said END symbol is transmitted on data lane 0.

26) The communication interface of claim 25 wherein each said data lane is identified by the alternating states of odd and even cycles.

27) The communication interface of claim 26 wherein said IDLE symbol is IDLE_EVEN during said even cycle and IDLE_ODD during said odd cycle.

28) The communication interface of claim 27 wherein all said data lane 0 through data lane n transmit IDLE_EVEN during said even cycles, and IDLE_ODD during said odd cycles.

SUB B1

29) The communication interface of claim 28 where IDLE_EVEN or IDLE_ODD are transmitted after said END symbol at least once during every interval $t_{elasticity}$.

30) The communication interface of claim 29 where $t_{elasticity} = T_{transmit} * clk\_offset$,

where

$T_{transmit}$ = time since last IDLE transmittal

clk_offset = (maximum Transmit clock rate - minimum receive clock rate)/(minimum receive clock rate).

31) A transmit processor comprising:

a busy input;

a transmit buffer/controller accepting packet data comprising a header and a payload as input, arranging said packet data into a plurality n of data lanes, and delivering to each said data lane unencoded transmit data and a control signal, whereby when said control signal is asserted, said unencoded transmit data includes at least one of the values START, END, IDLE, IDLE_BUSY and when said control signal is not asserted, said transmit data includes said packet data;

a plurality n of transmit encoders, each having an input and an output, each of said transmit encoder inputs uniquely coupled to one of said transmit buffer/controller data lanes, said transmit encoder input comprising said

unencoded transmit data and said control signal, said transmit encoder output producing a unique encoded output value for each said unencoded transmit data value when said control signal is not asserted, and producing a unique encoded output values for each unencoded transmit data START, END, IDLE, and IDLE_BUSY when said control signal is asserted;

a plurality n of transmit serializers, each having an input uniquely coupled to one of said transmit encoder outputs, said transmit serializers outputting a single serial stream of data from said transmit serializer input;

wherein said transmit buffer/controller sends said header by outputting on said first data lane the asserted said control and said unencoded transmit data START, and simultaneously outputs the remainder of said header on said remaining data lanes accompanied by said unasserted control signal for each said data lane,

thereafter and on each successive cycle said transmit buffer/controller distributes said payload data on all said data lanes and sends it to said transmit encoder with said unasserted control signal accompanied by said payload data, until unsent said payload data can not fully span said n data lanes,

thereafter said transmit buffer/controller sends the last said payload data on each said data lane with

associated said control signal unasserted, with following said data lane having said control signal asserted accompanied by said unencoded data END, and the remaining said data lanes having said control signal asserted accompanied by said unencoded data IDLE.

32) The transmit processor of claim 31 wherein each said transmit cycle has the state odd or even, and said IDLE comprises an IDLE_EVEN sent on said even cycles or an IDLE_ODD sent on said odd cycle.

33) The transmit processor of claim 32 wherein each successive transmit cycle alternates between odd or even, said IDLE_EVEN is sent during even cycles, and IDLE_ODD is sent during odd cycles.

34) The transmit processor of claim 32 wherein said IDLE comprises an IDLE when said busy input is not asserted, or a IDLE_BUSY when said busy input is asserted.

35) The transmit processor of claim 34 wherein said IDLE comprises an IDLE_EVEN_BUSY during said even cycle when said busy input is asserted, an IDLE_EVEN during said even cycle when said busy input is not asserted, an IDLE_ODD_BUSY during said odd cycle when said busy input is asserted, and

an IDLE_ODD during said odd cycle when said busy is not asserted.

36) The transmit processor of claim 35 wherein said transmit encoder comprises an 8B/10B encoder.

37) The transmit processor of claim 36 wherein the number of said data lanes n = 8.

38) The transmit processor of claim 36 wherein the number of said data lanes n = 4.

39) The transmit processor of claim 36 wherein the number of said data lanes n = 2.

40) The transmit processor of claim 36 wherein the number of said data lanes n = 1.

41) The transmit processor of claim 36 wherein the 10B coding value for symbol START is K27.7.

42) The transmit processor of claim 36 wherein the 10B coding value for symbol END is K29.7.

43) The transmit processor of claim 36 wherein the 10B coding value for symbol IDLE_EVEN is K28.5.

44) The transmit processor of claim 36 wherein the 10B coding value for symbol IDLE_ODD is K23.7.

45) The transmit processor of claim 36 wherein the 10B coding value for symbol IDLE_EVEN_BUSY is K28.1.

46) The transmit processor of claim 36 wherein the 10B coding value for symbol IDLE_ODD_BUSY is K28.0.

47) The transmit processor of claim 36 wherein the 10B coding values for the symbols START, END, IDLE_EVEN, IDLE_EVEN_BUSY, IDLE_ODD, and IDLE_ODD_BUSY have unique values when compared to any coded 10B data value.

48) The transmit processor of claim 47 wherein the 10B coding values for the symbols START, END, IDLE_EVEN, IDLE_EVEN_BUSY, IDLE_ODD, and IDLE_ODD_BUSY are separated by hamming distance 2.

49) A receive processor comprising:

a plurality n of receive deserializers each accepting as input a serial stream of encoded data and outputting deserialized encoded data;

a plurality n of receive decoders each uniquely coupled to and accepting as input said deserialized encoded data and providing as output decoded data and decoded control signals, said decoded data including at least on of the values START, END, and IDLE when said control signal is asserted;

a receive buffer/controller for the formation of data packets, said buffer/controller having a plurality n of inputs, each uniquely coupled to said decoded data and said decoded control, said buffer/controller having a busy output and a data output, said receive buffer/controller awaiting START on said first lane with associated control signal asserted, and storing a header on the remaining said data lanes when said START is received, and transferring to said data output all subsequent data while said control signal is unasserted for all said data lanes, and upon receipt of said END accompanied by the assertion of said associated control signal on any data lane, transferring said decoded data to said data output all said received data up to but not including said data lane having said control signal END.

50) The receive processor of claim 49 wherein said IDLE comprises the symbols IDLE_EVEN, IDLE_ODD, IDLE_EVEN_BUSY, and IDLE_EVEN_ODD.

51) The receive processor of claim 50 including a busy signal wherein the reception of IDLE_EVEN_BUSY or IDLE_ODD_BUSY causes said receive processor to assert said busy output.

52) The receive processor of claim 49 wherein said receive decoder uses a 10B/8B decoding method for converting said encoded data into said decoded data.

53) The receive processor of claim 52 wherein each receive deserializer achieves synchronization using the symbols IDLE_EVEN and IDLE_ODD.

54) the receive processor of claim 52 wherein the 10B coding value for symbol START is K27.7.

55) The receive processor of claim 52 wherein the 10B coding value for symbol END is K29.7.

56) The receive processor of claim 52 wherein the 10B coding value for symbol IDLE_EVEN is K28.5.

57) The receive processor of claim 52 wherein the 10B coding value for symbol IDLE_ODD is K23.7.

58) The receive processor of claim 52 wherein the 10B coding value for symbol IDLE_EVEN_BUSY is K28.1.

59) The receive processor of claim 52 wherein the 10B coding value for symbol IDLE_ODD_BUSY is K28.0.

60) The receive processor of claim 52 wherein the 10B encoded values for the symbols START, END, IDLE_EVEN and IDLE_ODD have unique values when compared to any other encoded 10B data value.

61) The receive processor of claim 60 wherein the 10B coding values for the symbols START, END, IDLE_EVEN, and IDLE_ODD are separated by hamming distance 2.

62) The receive processor of claim 49 wherein the number of data lanes n = 8.

63) The receive processor of claim 49 wherein the number of data lanes n = 4.

64) The receive processor of claim 49 wherein the number of data lanes n = 2.

65) The receive processor of claim 49 wherein the number of data lanes n = 1.

66) A communications interface for sending or receiving a packet, said packet comprising, in sequence, a header, variable length payload, and a terminator;

said header including a START symbol and a TYPE field identifying the format of said payload;

said terminator including an END symbol;

wherein said START symbol is transmitted first, followed by the remainder of said header, followed by said variable length packet data, followed by said terminator.

67) The interface of claim 66 wherein said TYPE field uniquely identifies said payload format, said format including Ethernet packets, ATM cells, and control packets.

68) the interface of claim 67 wherein said header further includes declaration fields for at least one of BPDU, PRIORITY, VLAN_ID, and an application specific field.

69) The interface of claim 68 wherein said BPDU field is 1 bit in size.

70) The interface of claim 69 wherein said PRIORITY field is 3 bits in size.

71) The interface of claim 70 wherein said VLAN_ID field is 12 bits in size.

72) The interface of claim 71 wherein said application specific field is 32 bits in size.